Metaprogramming on monoidal categories

Yuma Mizuno

January 17
Lean Together 2025

Outline

I will introduce two tools for monoidal categories, which are implemented by Lean's metaprogramming:

1. coherence tactic
2. string diagram widget

Outline

Metaprogramming for algebraic structures

- How to prove an equation in a monoid like

$$(a * (b * (1 * c)) * d) * e = (1 * (a * b)) * (c * (d * e))$$

  in Lean?

Metaprogramming for algebraic structures

- How to prove an equation in a monoid like

$$(a * (b * (1 * c)) * d) * e = (1 * (a * b)) * (c * (d * e))$$

  in Lean?
- Answer: simp only [mul_assoc, one_mul, mul_one]

Metaprogramming for algebraic structures

- How to prove an equation in a monoid like

$$(a * (b * (1 * c)) * d) * e = (1 * (a * b)) * (c * (d * e))$$

  in Lean?
- Answer: `simp only [mul_assoc, one_mul, mul_one]`
- "Meta theorem": any such equation can be simplified to

$$a * b * c * d * e$$

  by iterated application of the associative law and the unit laws.

Metaprogramming for algebraic structures

- In other situations, we often need specialized tactics (ring, abel, etc.).
- The coherence tactic proves coherence conditions in monoidal categories.

Monoidal categories

A monoidal category consists of

- A : category
- $\otimes : A \times A \to A$
- $1 : A$
- associator $\alpha : \forall a, b, c : A,\ (a \otimes b) \otimes c \cong a \otimes (b \otimes c)$
- left unitor $\lambda : \forall a : A,\ 1 \otimes a \cong a$
- right unitor $\rho : \forall a : A,\ a \otimes 1 \cong a$
- coherence conditions (next slide)

Coherence conditions

- We can define an isomorphism with the following type using associator and unitors:

$$(a \otimes (b \otimes (1 \otimes c)) \otimes d) \otimes e \cong (1 \otimes (a \otimes b)) \otimes (c \otimes (d \otimes e)),$$

- but there are several ways to do so.
- Coherence conditions state that they are equal.

Coherence theorem

All coherence conditions follow from the following two conditions:

- pentagon identity:

$$\begin{tikzcd}
 & ((a \otimes b) \otimes c) \otimes d \arrow[dl, "\alpha \otimes 1"'] \arrow[dr, "\alpha"] & \\
(a \otimes (b \otimes c)) \otimes d \arrow[d, "\alpha"'] & & (a \otimes b) \otimes (c \otimes d) \arrow[d, "\alpha"] \\
a \otimes ((b \otimes c) \otimes d) \arrow[rr, "1 \otimes \alpha"'] & & a \otimes (b \otimes (c \otimes d))
\end{tikzcd}$$

- triangle identity:

$$\begin{tikzcd}
 & (a \otimes 1) \otimes b \arrow[dl, "\alpha"'] \arrow[dr, "\rho \otimes 1"] & \\
a \otimes (1 \otimes b) \arrow[rr, "1 \otimes \lambda"'] & & a \otimes b
\end{tikzcd}$$

Monoidal categories

A monoidal category consists of

- A : category
- $\otimes : A \times A \to A$
- $1 : A$
- associator $\alpha : \forall a, b, c : A,\ (a \otimes b) \otimes c \cong a \otimes (b \otimes c)$
- left unitor $\lambda : \forall a : A,\ 1 \otimes a \cong a$
- right unitor $\rho : \forall a : A,\ a \otimes 1 \cong a$
- pentagon and triangle identities

Coherence tactic

- How to prove coherence conditions in Lean?

Coherence tactic

- How to prove coherence conditions in Lean?
- Answer: coherence tactic

Coherence tactic

There are two ways to implement the coherence tactic:

1. Transporting an equation from a formalized coherence theorem, which is a theorem about free monoidal categories (formalized in Lean by Markus Himmel).
2. Direct proof-producing tactic, manipulating Lean.Expr terms.

Experience has shown that

- the transporting tactic (at least without an optimization effort) relies on heavy definitional equalities, and is very slow when expressions are long,
- the proof-producing tactic works well in practice.

Mathlib previously used the first method, but recently switched to the second method.

Demo

- monoidal tactic in mathlib
- set_option trace.monoidal true to see proof steps

Expression types

When manipulating Lean expressions, it is useful to define specialized expression types. We do not use dependent types.

```
inductive Obj : Type
| unit (e : Expr) : Obj
| tensor (e : Expr) (X Y : Obj) : Obj
| of (e : Expr) : Obj

inductive Iso : Type
| id (e : Expr) (X : Obj) : Iso
| associator (e : Expr) (X Y Z : Obj) : Iso
| leftUnitor (e : Expr) (X : Obj) : Iso
| rightUnitor (e : Expr) (X : Obj) : Iso
| comp (e : Expr) (X Y Z : Obj) (f g : Iso) : Iso
| tensor (e : Expr) (X₁ Y₁ X₂ Y₂ : Obj) (f g : Iso) : Iso
| inv (e : Expr) (X Y : Obj) (f : Iso) : Iso
| of (e : Expr) (X Y : Obj) : Iso
```

Metaprogramming TIPS

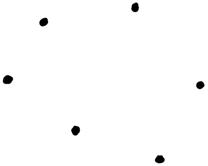
```
def tensorM (X Y : Obj) : MonoidalM Obj := do
  let ctx ← read
  let .some _monoidal := ctx.instMonoidal? | synthMonoidalError
  let X_e : Q($ctx.C) := X.e
  let Y_e : Q($ctx.C) := Y.e
  return .tensor q($X_e ⊗ $Y_e) X Y
```

- Monads are always useful.
- Qq is useful when constructing Lean.Expr terms.
- Some MetaM functions (synthInstance, mkAppM, etc.) are not sufficiently fast for our purpose.

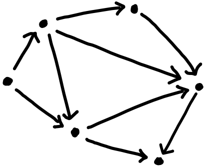
Outline

2-categorical point of view

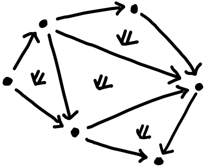
n	n-category	one-object case
0	set	point
1	category	monoid
2	bicategory	monoidal category



$n = 0$



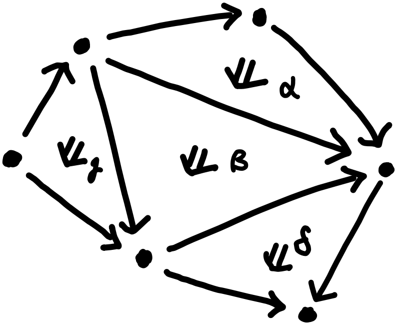
$n = 1$

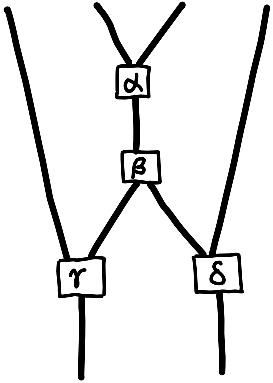


$n = 2$

String diagram

- String diagrams are the dual of 2-cell diagrams.





Demo

- #string_diagram command
- with_panel_widgets [Mathlib.Tactic.Widget.StringDiagram] in tactic proofs

Implementation of string diagram widget

- Normalization of for 2-morphisms
  - Lean.Expr terms manipulation
  - extracting "atomic" 2-morphisms and removing associators and unitors
- Drawing by ProofWidgets [Nawrocki et.al.], with its Penrose support
  - Penrose is a software for drawing diagrams by specifying constraints.

Remark

- The monoidal and bicategory tactics are general-purpose tactics for proving equations involving 2-morphisms. They solve any equation such that the LHS and the RHS have the same string diagram.
- Normalization of for 2-morphisms for drawing string diagram is actually a part of these tactics.

Future directions

- more general-purpose tactic, which involves the whisker exchange relation
- coherence tactic for symmetric monoidal categories
- string diagrams to codes
- drawing pasting diagrams